Final Project Submission - Module 02

- Student name: Maria Antonietta Ricci
- Student pace: Data Science On line - Self-paced
- Instructor name: Jeff
- Blog post URL: https://datastorytell.wordpress.com/2022/05/17/thomas-bayes-his-time-his-theorem/

Business Problem

This analysis aims to establish how home renovations might increase the estimated value of their homes, and by what amount. Stakeholders who could benefit having this information are private homeowners, investors, renovators and real estate agencies and buyers as well. Renovating a house can be considered an interesting alternative when it comes to acquire a property in the real estate market. It can constitute, among other advantages, a lucrative purchase as, after renovation, the property can be sold at a higher price, with a relative important profitable margin.

Hypothesis 1

Y(price) = Bo + B1(renovation) + e1

If B1 is positive, Y increases. If B1 is negative, Y decreases. If B1 is = 0, no effect. If B1 != 0, effect = H1.

H1: Price is impacted by House Renovation.

if B1 = 0, no effect = H0

H0: Price is not impacted by House Renovation.

In [1]:
```
# Installing needed libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import scipy
from scipy import stats
```

The dataset chosen for this analysis: King County House Sales

In [2]:
```
# Importing and reading dataset
os.chdir("/Users/mariaantoniettaricci/DSC/Module_02/Final_Project/dsc-phase-2-project/
kc_house = pd.read_csv("kc_house_data.csv")
kc_house.head()
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vi
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.0	
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	

5 rows × 21 columns

In [3]:
```
#Cleaning the dataset
kc_house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21597 non-null  int64
 1   date           21597 non-null  object
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64
 6   sqft_lot       21597 non-null  int64
 7   floors         21597 non-null  float64
 8   waterfront     19221 non-null  float64
 9   view           21534 non-null  float64
 10  condition      21597 non-null  int64
 11  grade          21597 non-null  int64
 12  sqft_above     21597 non-null  int64
 13  sqft_basement  21597 non-null  object
 14  yr_built       21597 non-null  int64
 15  yr_renovated   17755 non-null  float64
 16  zipcode        21597 non-null  int64
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64
 20  sqft_lot15     21597 non-null  int64
dtypes: float64(8), int64(11), object(2)
memory usage: 3.5+ MB
```

By all means this is indeed a very large dataset: 21957 entries, 21 columns.

In [4]:
```
kc_house.columns
```

Out[4]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

Description of the labels as follows:

Column Names and descriptions for Kings County Data Set

- **id** - unique identified for a house
- **date** - house was sold
- **price** - is prediction target
- **bedrooms** - of Bedrooms/House
- **bathrooms** - of bathrooms/bedrooms
- **sqft_living** - footage of the home
- **sqft_lot** - footage of the lot
- **floors** - floors (levels) in house
- **waterfront** - House which has a view to a waterfront
- **view** - Has been viewed
- **condition** - How good the condition is (Overall)
- **grade** - overall grade given to the housing unit, based on King County grading system
- **sqft_above** - square footage of house apart from basement
- **sqft_basement** - square footage of the basement
- **yr_built** - Built Year
- **yr_renovated** - Year when house was renovated
- **zipcode** - zip
- **lat** - Latitude coordinate
- **long** - Longitude coordinate
- **sqft_living15** - The square footage of interior housing living space for the nearest 15 neighbors
- **sqft_lot15** - The square footage of the land lots of the nearest 15 neighbors

In [5]:
```
#Rename the columns in order to make them more clear and intuitive
kc_house.rename(columns ={"date": "sell date", "grade": "WSS grade"})
```

Out[5]:

	id	sell date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfror
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	Na
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.
...	...	...	...	...	...	...	...	...	...
21592	263000018	5/21/2014	360000.0	3	2.50	1530	1131	3.0	0.
21593	6600060120	2/23/2015	400000.0	4	2.50	2310	5813	2.0	0.
21594	1523300141	6/23/2014	402101.0	2	0.75	1020	1350	2.0	0.
21595	291310100	1/16/2015	400000.0	3	2.50	1600	2388	2.0	Na
21596	1523300157	10/15/2014	325000.0	2	0.75	1020	1076	2.0	0.

21597 rows × 21 columns

In [6]:
```
#Checking for missing values
kc_house.isna().sum()
```

Out[6]:
```
id                  0
date                0
price               0
bedrooms            0
bathrooms           0
sqft_living         0
sqft_lot            0
floors              0
waterfront       2376
view               63
condition           0
grade               0
sqft_above          0
sqft_basement       0
yr_built            0
yr_renovated     3842
zipcode             0
lat                 0
long                0
sqft_living15       0
sqft_lot15          0
dtype: int64
```

In [7]:
```
kc_house["grade"].unique()
```

Out[7]:
```
array([ 7,  6,  8, 11,  9,  5, 10, 12,  4,  3, 13])
```

In [8]:
```
wrf = kc_house["waterfront"].unique()
wrf
```

Out[8]:
```
array([nan,  0.,  1.])
```

Looking at the dataset, some considerations emerge:

- further research: trend has a seasonal/time base, and zipcodde (geographical base).
- difference between the grading and King County evaluation system (any relationship can be found between the two evaluation systems (needed to clarify the meaning).
- relationship between yr_built and yr_renovation.
- relationship between yr_built and yr_renovation, and two evaluation systems.
- relationships between sqft_living, on one hand, and sqft_above, sqft_basement, on the other (the two latter seemed to be implied in the former).
- presence of bedrooms and bathrooms what implies?
- Three columns appear to have missing values in form of nan and "?":
- Column "view": how many times the house (or the postings?) has been viewed. This column seem to be not relevant for the purpose of this study so I decided drop it with a certain degree of confidence.
- The other two columns evaluation is more complex as:
  - Column "waterfront" and Column "yr_renovated" is a feature that impacts the value of a house. I can't substitute the values in "waterfront" with the mean since we have binary values: either the house has a waterfront or doesn't.
  - Column "yr_renovation". Categorical data about dates. It is not possible to use the mean of the column values.

I can't really use the data that don't have this information. That's the reason I decided to drop it, even though unfortunately, I lose 1/3 of the data.

In [9]:
```
kch2 = kc_house.copy()
kch2
```

Out[9]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfror
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	Na
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.
...	...	...	...	...	...	...	...	...	...
21592	263000018	5/21/2014	360000.0	3	2.50	1530	1131	3.0	0.
21593	6600060120	2/23/2015	400000.0	4	2.50	2310	5813	2.0	0.
21594	1523300141	6/23/2014	402101.0	2	0.75	1020	1350	2.0	0.
21595	291310100	1/16/2015	400000.0	3	2.50	1600	2388	2.0	Na
21596	1523300157	10/15/2014	325000.0	2	0.75	1020	1076	2.0	0.

21597 rows × 21 columns

In [10]:
```
kc_house.drop(["view"], axis=1, inplace = True)
```

In [11]:
```
kc_house.dropna(inplace= True)
kc_house
```

Out[11]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfro
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	(
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	(
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	(
5	7237550310	5/12/2014	1230000.0	4	4.50	5420	101930	1.0	(
6	1321400060	6/27/2014	257500.0	3	2.25	1715	6819	2.0	(
...	...	...	...	...	...	...	...	...	
21591	2997800021	2/19/2015	475000.0	3	2.50	1310	1294	2.0	(
21592	263000018	5/21/2014	360000.0	3	2.50	1530	1131	3.0	(
21593	6600060120	2/23/2015	400000.0	4	2.50	2310	5813	2.0	(
21594	1523300141	6/23/2014	402101.0	2	0.75	1020	1350	2.0	(
21596	1523300157	10/15/2014	325000.0	2	0.75	1020	1076	2.0	(

15809 rows × 20 columns

In [12]:
```
kc_house.drop(index=kc_house[kc_house['sqft_basement'] == "?"].index, inplace=True)
kc_house
```

Out[12]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfro
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	(
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	(
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	(
5	7237550310	5/12/2014	1230000.0	4	4.50	5420	101930	1.0	(
7	2008000270	1/15/2015	291850.0	3	1.50	1060	9711	1.0	(
...	...	...	...	...	...	...	...	...	
21591	2997800021	2/19/2015	475000.0	3	2.50	1310	1294	2.0	(
21592	263000018	5/21/2014	360000.0	3	2.50	1530	1131	3.0	(
21593	6600060120	2/23/2015	400000.0	4	2.50	2310	5813	2.0	(
21594	1523300141	6/23/2014	402101.0	2	0.75	1020	1350	2.0	(
21596	1523300157	10/15/2014	325000.0	2	0.75	1020	1076	2.0	(

15474 rows × 20 columns

In [13]:
```
#Adjusting the data types in order to be able to work on them effectively.
#object: id,date,waterfront,built,renovated,zipcode
#float:price,bedrooms,bathrooms,sqft_living,sqft_lot,floors,condition,grade,sqft_above
# lat,lon,l15,t15
kc_house[['id', 'date',"yr_built", "yr_renovated"]] = kc_house[['id', 'date',"yr_built
kc_house[["price","bedrooms","bathrooms", "sqft_living","sqft_lot","waterfront", "zipc
```

In [14]:
```
kc_house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15474 entries, 1 to 21596
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             15474 non-null  object
 1   date           15474 non-null  object
 2   price          15474 non-null  float64
 3   bedrooms       15474 non-null  float64
 4   bathrooms      15474 non-null  float64
 5   sqft_living    15474 non-null  float64
 6   sqft_lot       15474 non-null  float64
 7   floors         15474 non-null  float64
 8   waterfront     15474 non-null  float64
 9   condition      15474 non-null  float64
 10  grade          15474 non-null  float64
 11  sqft_above     15474 non-null  float64
 12  sqft_basement  15474 non-null  float64
 13  yr_built       15474 non-null  object
 14  yr_renovated   15474 non-null  object
 15  zipcode        15474 non-null  float64
 16  lat            15474 non-null  float64
 17  long           15474 non-null  float64
 18  sqft_living15  15474 non-null  float64
 19  sqft_lot15     15474 non-null  float64
dtypes: float64(16), object(4)
memory usage: 2.5+ MB
```

After cleaning, the dataset is still pretty huge with 15474 entries and 19 columns, both numerical and categorical as the datatype indicates.

EDA & FEATURE ENGINEERING

In [15]:
```
kc_house.describe()
```

Out[15]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wate
count	1.547400e+04	15474.000000	15474.000000	15474.000000	1.547400e+04	15474.000000	15474.0(
mean	5.417604e+05	3.379088	2.121559	2086.045302	1.530465e+04	1.494378	0.0
std	3.747690e+05	0.934149	0.767421	920.447871	4.206225e+04	0.538777	0.08
min	8.200000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	0.0(
25%	3.200000e+05	3.000000	1.750000	1430.000000	5.057000e+03	1.000000	0.0(
50%	4.500000e+05	3.000000	2.250000	1920.000000	7.620000e+03	1.500000	0.0(
75%	6.434758e+05	4.000000	2.500000	2550.000000	1.071375e+04	2.000000	0.0(
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.0(

Checking max and min, I found some outliers. I exclude them based on the fact that they may not be considered the average single family house, shrinking the spectrum of potential stakeholders. Houses with more than 5000 square feet are considering mansions.

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjG_pn134_3AhUOLDQIHSbiBV is-a-mansion/&usg=AOvVaw1bEtkLfEA3ss_W4JxzmH1a

In [16]:
```
kc_house = kc_house.loc[kc_house["sqft_living"] < 5000.0]
```

In [17]:
```
kc_house
```

Out[17]:

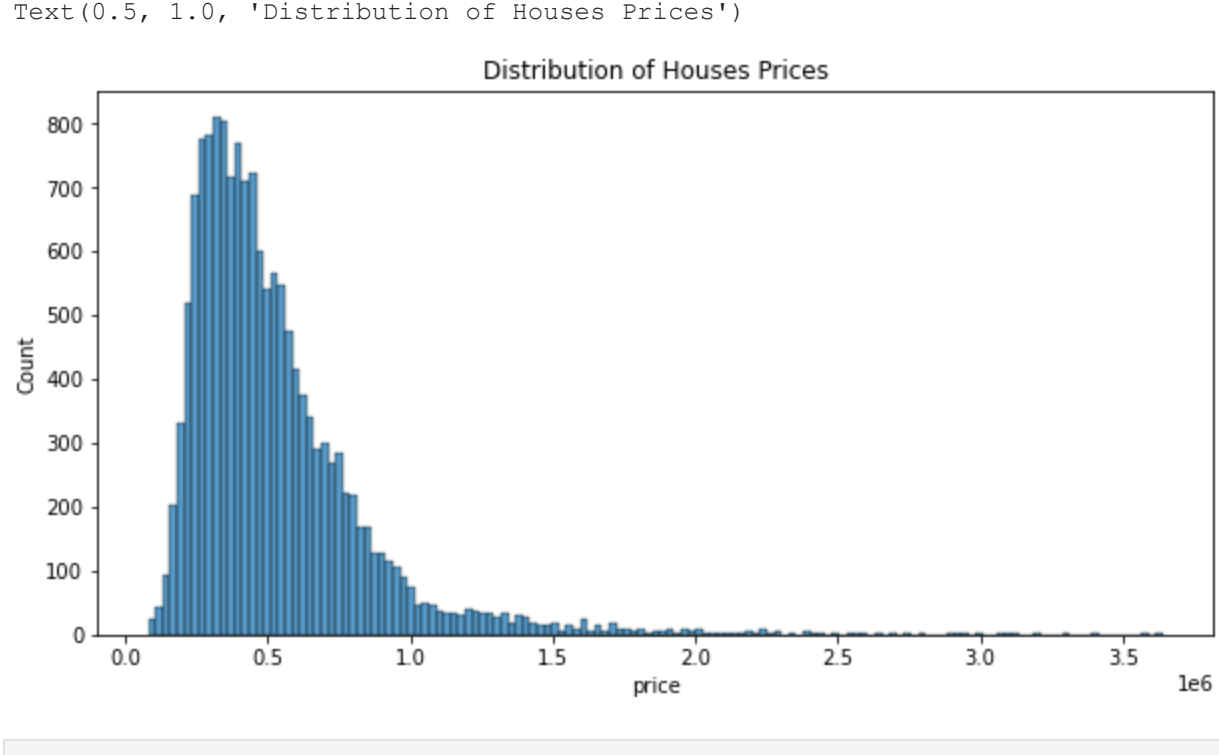
	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfror
1	6414100192	12/9/2014	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.
3	2487200875	12/9/2014	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.
4	1954400510	2/18/2015	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.
7	2008000270	1/15/2015	291850.0	3.0	1.50	1060.0	9711.0	1.0	0.
8	2414600126	4/15/2015	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.
...	...	...	...	...	...	...	...	...	...
21591	2997800021	2/19/2015	475000.0	3.0	2.50	1310.0	1294.0	2.0	0.
21592	263000018	5/21/2014	360000.0	3.0	2.50	1530.0	1131.0	3.0	0.
21593	6600060120	2/23/2015	400000.0	4.0	2.50	2310.0	5813.0	2.0	0.
21594	1523300141	6/23/2014	402101.0	2.0	0.75	1020.0	1350.0	2.0	0.
21596	1523300157	10/15/2014	325000.0	2.0	0.75	1020.0	1076.0	2.0	0.

15323 rows × 20 columns

Visualizing the dependent variable.

In [18]:
```
data = kc_house["price"]
ax = plt.figure(figsize=(10, 5))
sns.histplot(data = data)
plt.title('Distribution of Houses Prices')
```

Out[18]:
```
Text(0.5, 1.0, 'Distribution of Houses Prices')
```



In [19]:
```
import warnings
kch_price = kc_house["price"]
warnings.filterwarnings(action='ignore', category=FutureWarning)
sns.distplot(kch_price, bins=None, hist=True, kde=True, rug=False, fit=None, hist_kws=
plt.title('Distribution of Houses Prices')
```

Out[19]:
```
Text(0.5, 1.0, 'Distribution of Houses Prices')
```

In [20]:
```
data.describe()
```

Out[20]:
```
count    1.532300e+04
mean     5.257012e+05
std      3.164254e+05
min      8.200000e+04
25%      3.200000e+05
50%      4.500000e+05
75%      6.350000e+05
max      3.640000e+06
Name: price, dtype: float64
```

The plot shows:

- skeweness
- kurtosis
- non normally distributed

Checking again for other outliers calcutating z-values.

In [ ]:

In [21]:
```
# Calculating z scores and filtering the outliers (observations 3 std deviations away
a = list(scipy.stats.zscore(kc_house["price"]))
a
kc_hs = kc_house.assign(pzs = a)
kc_hs = kc_hs[kc_hs["pzs"]<= 3]
kc_hs
```

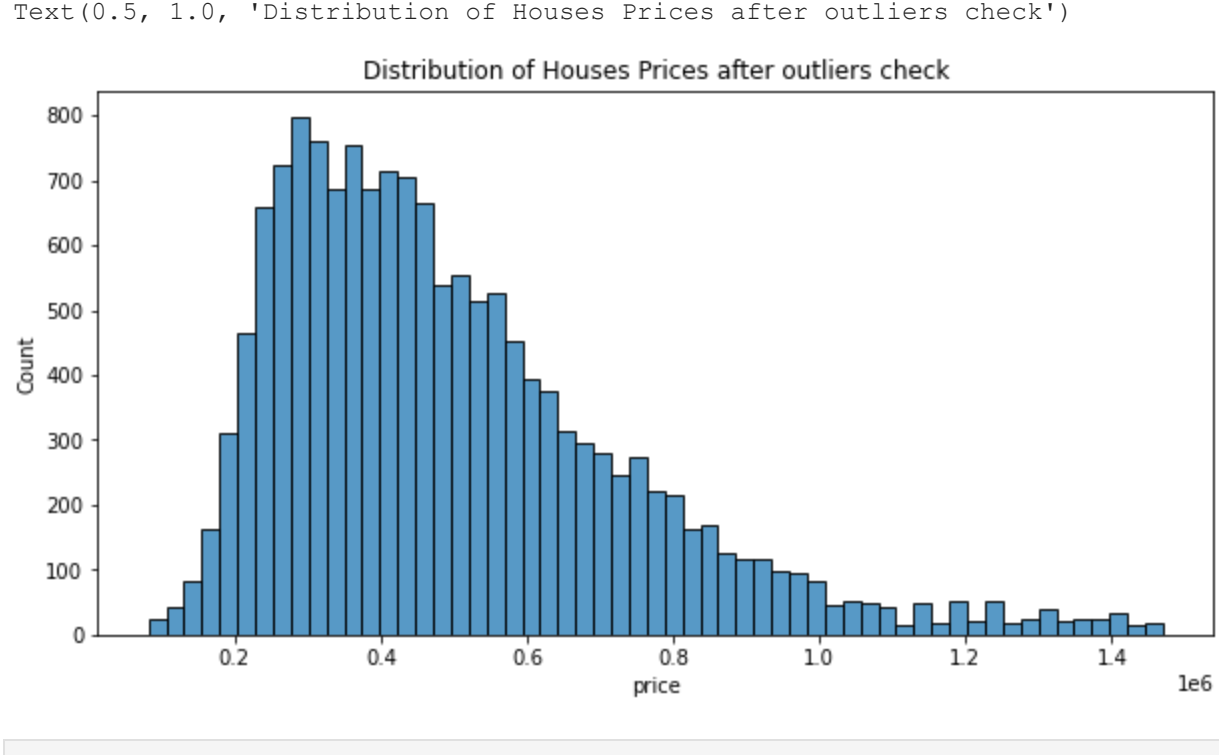
Out[21]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfror
1	6414100192	12/9/2014	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.
3	2487200875	12/9/2014	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.
4	1954400510	2/18/2015	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.
7	2008000270	1/15/2015	291850.0	3.0	1.50	1060.0	9711.0	1.0	0.
8	2414600126	4/15/2015	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.
...	...	...	...	...	...	...	...	...	...
21591	2997800021	2/19/2015	475000.0	3.0	2.50	1310.0	1294.0	2.0	0.
21592	263000018	5/21/2014	360000.0	3.0	2.50	1530.0	1131.0	3.0	0.
21593	6600060120	2/23/2015	400000.0	4.0	2.50	2310.0	5813.0	2.0	0.
21594	1523300141	6/23/2014	402101.0	2.0	0.75	1020.0	1350.0	2.0	0.
21596	1523300157	10/15/2014	325000.0	2.0	0.75	1020.0	1076.0	2.0	0.

15029 rows × 21 columns

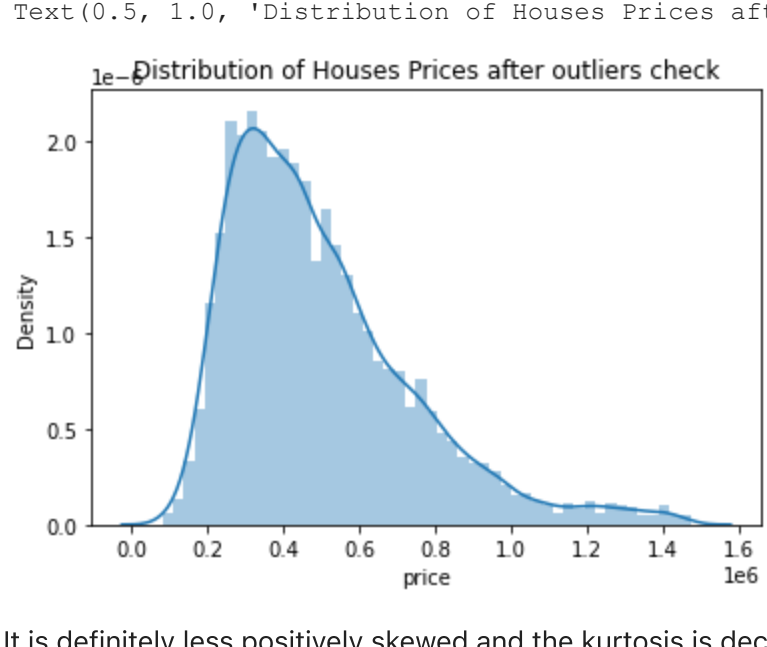
In [22]:
```
data= kc_hs["price"]
ax = plt.figure(figsize=(10, 5))
sns.histplot(data = data)
plt.title('Distribution of Houses Prices after outliers check')
```

Out[22]:
```
Text(0.5, 1.0, 'Distribution of Houses Prices after outliers check')
```



In [23]:
```
kch_price = kc_hs["price"]
warnings.filterwarnings(action='ignore', category=FutureWarning)
sns.distplot(kch_price, bins=None, hist=True, kde=True, rug=False, fit=None, hist_kws=
plt.title('Distribution of Houses Prices after outliers check')
```

Out[23]:
```
Text(0.5, 1.0, 'Distribution of Houses Prices after outliers check')
```



It is definitely less positively skewed and the kurtosis is decreased as well. It is still not strictly bell shaped but it is good enough.

In [24]:
```
kc_hs
```

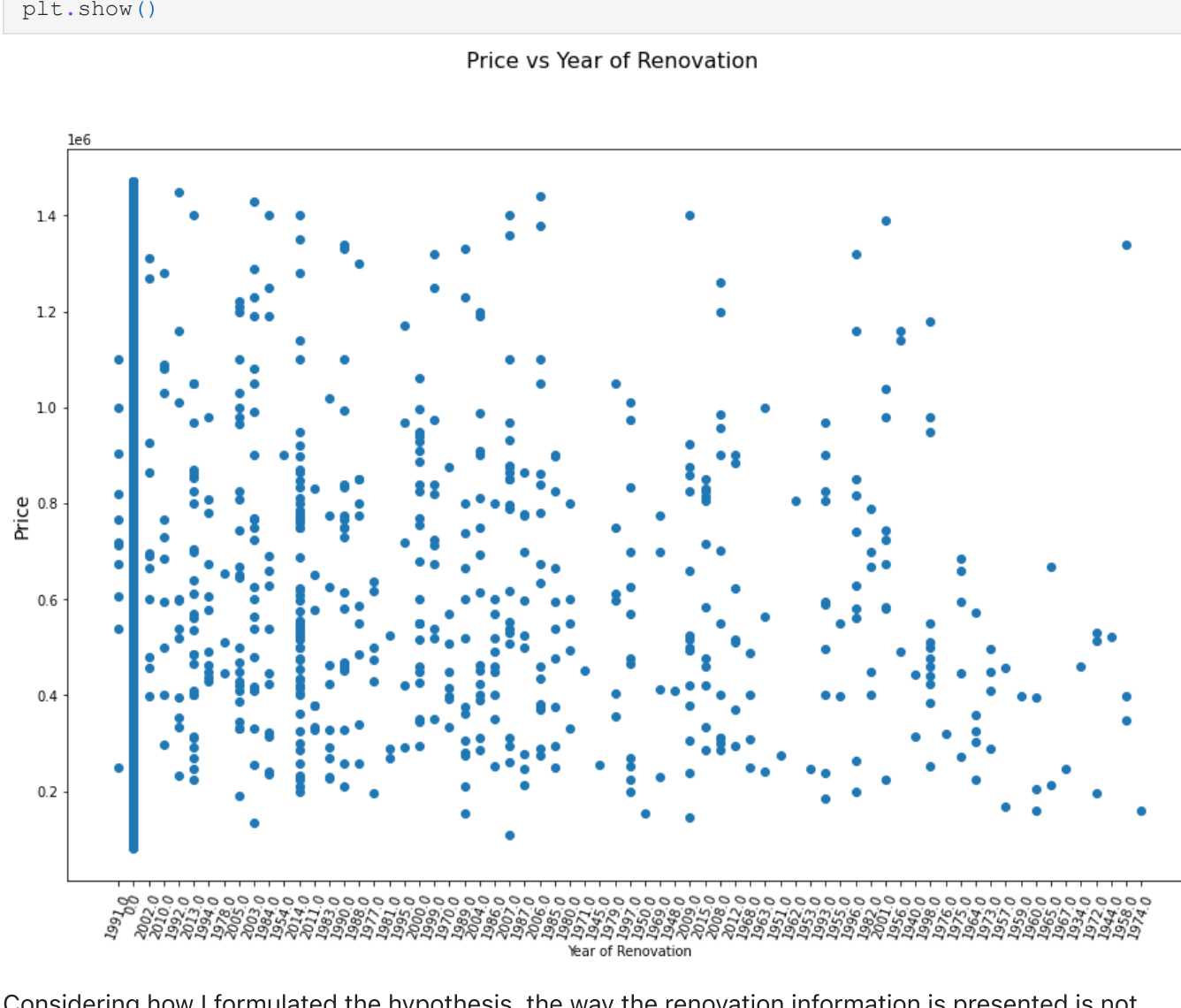
Out[24]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfror
1	6414100192	12/9/2014	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.
3	2487200875	12/9/2014	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.
4	1954400510	2/18/2015	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.
7	2008000270	1/15/2015	291850.0	3.0	1.50	1060.0	9711.0	1.0	0.
8	2414600126	4/15/2015	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.
...	...	...	...	...	...	...	...	...	...
21591	2997800021	2/19/2015	475000.0	3.0	2.50	1310.0	1294.0	2.0	0.
21592	263000018	5/21/2014	360000.0	3.0	2.50	1530.0	1131.0	3.0	0.
21593	6600060120	2/23/2015	400000.0	4.0	2.50	2310.0	5813.0	2.0	0.
21594	1523300141	6/23/2014	402101.0	2.0	0.75	1020.0	1350.0	2.0	0.
21596	1523300157	10/15/2014	325000.0	2.0	0.75	1020.0	1076.0	2.0	0.

15029 rows × 21 columns

Considering the hypothesis, I proceed visualizing the data and check relationships between price and renovation. As stated in the business problem, this is the main relationship I'm especilly interested in.

In [25]:
```
fig = plt.figure(figsize=(15, 10))
fig.suptitle('Price vs Year of Renovation', fontsize=16)
plt.scatter(data = kc_hs, x = "yr_renovated", y = "price")
plt.xlabel('Year of Renovation', fontsize=10)
plt.ylabel('Price', fontsize=14)
plt.xticks(rotation=70)
plt.show()
```



Considering how I formulated the hypothesis, the way the renovation information is presented is not usable. I decided to convert the related categorical values in binary values.

In [26]:
```
# converting to binary data
kc_hs["renovation"] = kc_hs["yr_renovated"]
kc_hs
```

Out[26]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfror
1	6414100192	12/9/2014	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.
3	2487200875	12/9/2014	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.
4	1954400510	2/18/2015	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.
7	2008000270	1/15/2015	291850.0	3.0	1.50	1060.0	9711.0	1.0	0.
8	2414600126	4/15/2015	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.
...	...	...	...	...	...	...	...	...	...
21591	2997800021	2/19/2015	475000.0	3.0	2.50	1310.0	1294.0	2.0	0.
21592	263000018	5/21/2014	360000.0	3.0	2.50	1530.0	1131.0	3.0	0.
21593	6600060120	2/23/2015	400000.0	4.0	2.50	2310.0	5813.0	2.0	0.
21594	1523300141	6/23/2014	402101.0	2.0	0.75	1020.0	1350.0	2.0	0.
21596	1523300157	10/15/2014	325000.0	2.0	0.75	1020.0	1076.0	2.0	0.

15029 rows × 22 columns

The reason I chose the following code string instead of using the function get_dummies() is because it would have reversed the attribution of the dummy variables would be inverted. Thus, proceeding with OLS analysis, it would e necessary to change the signs of model as well.

In [27]:
```
kc_hs["renovation"] = np.where(kc_hs["renovation"] != "0.0", 1, 0)
kc_hs
```

Out[27]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfror
1	6414100192	12/9/2014	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.
3	2487200875	12/9/2014	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.
4	1954400510	2/18/2015	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.
7	2008000270	1/15/2015	291850.0	3.0	1.50	1060.0	9711.0	1.0	0.
8	2414600126	4/15/2015	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.
...	...	...	...	...	...	...	...	...	...
21591	2997800021	2/19/2015	475000.0	3.0	2.50	1310.0	1294.0	2.0	0.
21592	263000018	5/21/2014	360000.0	3.0	2.50	1530.0	1131.0	3.0	0.
21593	6600060120	2/23/2015	400000.0	4.0	2.50	2310.0	5813.0	2.0	0.
21594	1523300141	6/23/2014	402101.0	2.0	0.75	1020.0	1350.0	2.0	0.
21596	1523300157	10/15/2014	325000.0	2.0	0.75	1020.0	1076.0	2.0	0.

15029 rows × 22 columns

In [28]:
```
kcc_hss = kc_hs.copy()
kcc_hss
```

Out[28]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfror
1	6414100192	12/9/2014	538000.0	3.0	2.25	2570.0	7242.0	2.0	0.
3	2487200875	12/9/2014	604000.0	4.0	3.00	1960.0	5000.0	1.0	0.
4	1954400510	2/18/2015	510000.0	3.0	2.00	1680.0	8080.0	1.0	0.
7	2008000270	1/15/2015	291850.0	3.0	1.50	1060.0	9711.0	1.0	0.
8	2414600126	4/15/2015	229500.0	3.0	1.00	1780.0	7470.0	1.0	0.
...	...	...	...	...	...	...	...	...	...
21591	2997800021	2/19/2015	475000.0	3.0	2.50	1310.0	1294.0	2.0	0.
21592	263000018	5/21/2014	360000.0	3.0	2.50	1530.0	1131.0	3.0	0.
21593	6600060120	2/23/2015	400000.0	4.0	2.50	2310.0	5813.0	2.0	0.
21594	1523300141	6/23/2014	402101.0	2.0	0.75	1020.0	1350.0	2.0	0.
21596	1523300157	10/15/2014	325000.0	2.0	0.75	1020.0	1076.0	2.0	0.

15029 rows × 22 columns

In [29]:
```
#Log Trasformation
kcc_hss["price"] = np.log(kcc_hss["price"])
kcc_hss
```

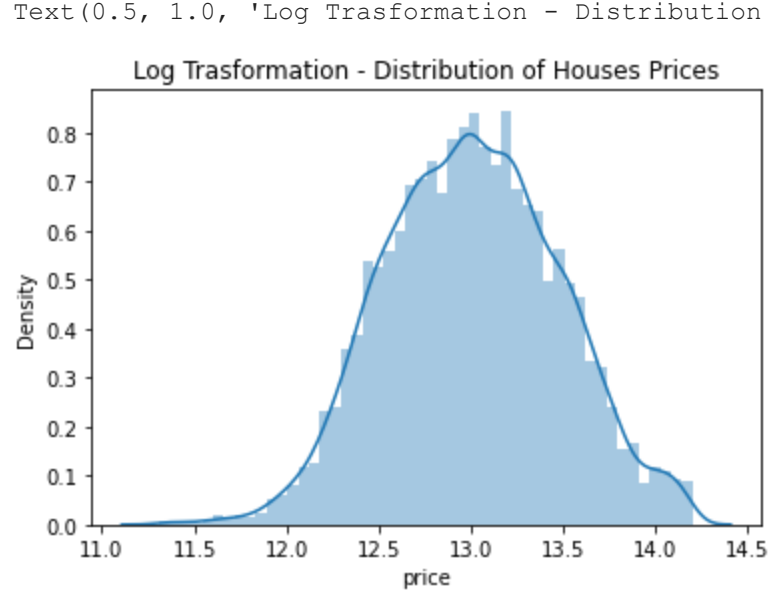
Out[29]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfrc
1	6414100192	12/9/2014	13.195614	3.0	2.25	2570.0	7242.0	2.0	(
3	2487200875	12/9/2014	13.311329	4.0	3.00	1960.0	5000.0	1.0	(
4	1954400510	2/18/2015	13.142166	3.0	2.00	1680.0	8080.0	1.0	(
7	2008000270	1/15/2015	12.583995	3.0	1.50	1060.0	9711.0	1.0	(
8	2414600126	4/15/2015	12.343658	3.0	1.00	1780.0	7470.0	1.0	(
...	...	...	...	...	...	...	...	...	
21591	2997800021	2/19/2015	13.071070	3.0	2.50	1310.0	1294.0	2.0	(
21592	263000018	5/21/2014	12.793859	3.0	2.50	1530.0	1131.0	3.0	(
21593	6600060120	2/23/2015	12.899220	4.0	2.50	2310.0	5813.0	2.0	(
21594	1523300141	6/23/2014	12.904459	2.0	0.75	1020.0	1350.0	2.0	(
21596	1523300157	10/15/2014	12.691580	2.0	0.75	1020.0	1076.0	2.0	(

15029 rows × 22 columns

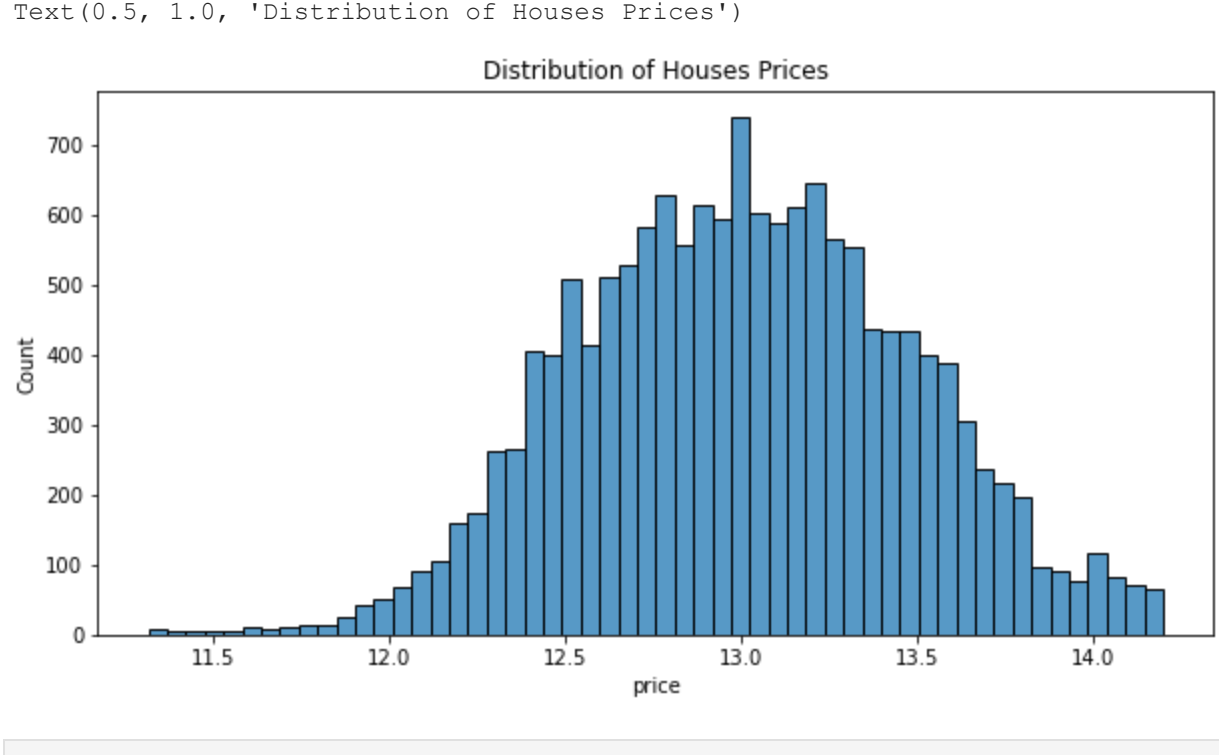
In [30]:
```
kcchs_price = kcc_hss["price"]
warnings.filterwarnings(action='ignore', category=FutureWarning)
sns.distplot(kcchs_price, bins=None, hist=True, kde=True, rug=False, fit=None, hist_kw
plt.title("Log Trasformation - Distribution of Houses Prices")
```

Out[30]:
```
Text(0.5, 1.0, 'Log Trasformation - Distribution of Houses Prices')
```



In [31]:
```
data2 = kcchs_price
ax = plt.figure(figsize=(10, 5))
sns.histplot(data = data2)
plt.title('Distribution of Houses Prices')
```

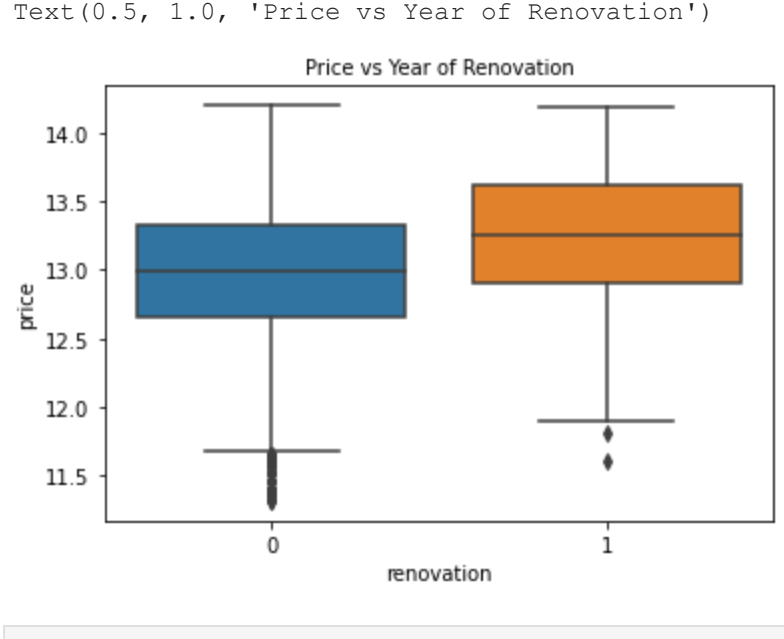
Out[31]:
```
Text(0.5, 1.0, 'Distribution of Houses Prices')
```



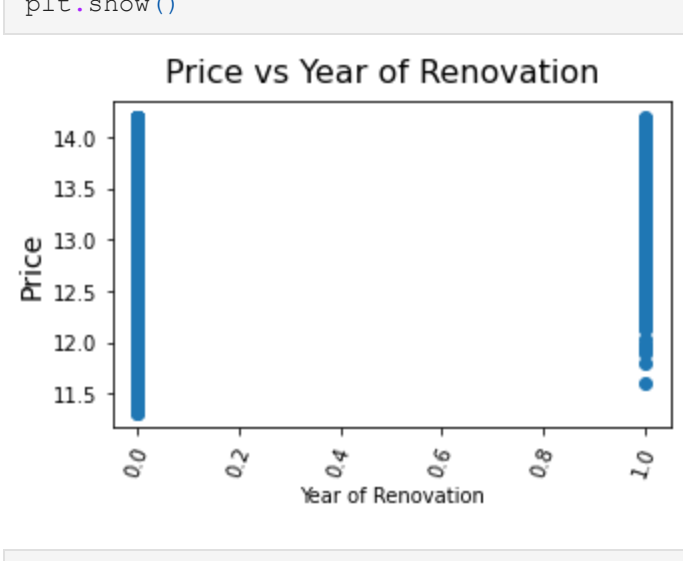
In [32]:
```
kcc_hss10 = kcc_hss[["price", "renovation"]]
```

In [52]:
```
ax = sns.boxplot(x = "renovation", y = "price", data = kcc_hss10)
ax.set_title('Price vs Year of Renovation', fontsize=10)
```

Out[52]:
```
Text(0.5, 1.0, 'Price vs Year of Renovation')
```



In [34]:
```
fig = plt.figure(figsize=(5, 3))
fig.suptitle('Price vs Year of Renovation', fontsize=16)
plt.scatter(data = kcc_hss10, x = "renovation", y = "price")
plt.xlabel('Year of Renovation', fontsize=10)
plt.ylabel('Price', fontsize=14)
plt.xticks(rotation=70)
plt.show()
```



In [35]:
```
# Running the regression Ln(y(price))= b0 + b1(renovation) + e1
from statsmodels.formula.api import ols
```

In [36]:
```
outcome = 'price'
x_cols = ['renovation']
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
```

```python
model = ols(formula=formula, data=kcc_hss).fit()
model.summary()
```

Out[36]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.009
Model:	OLS	**Adj. R-squared:**	0.009
Method:	Least Squares	**F-statistic:**	131.4
Date:	Mon, 20 Jun 2022	**Prob (F-statistic):**	2.67e-30
Time:	22:04:58	**Log-Likelihood:**	-9966.4
No. Observations:	15029	**AIC:**	1.994e+04
Df Residuals:	15027	**BIC:**	1.995e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	12.9994	0.004	3326.869	0.000	12.992	13.007
renovation	0.2278	0.020	11.464	0.000	0.189	0.267

Omnibus:	48.812	**Durbin-Watson:**	1.970
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	37.703
Skew:	-0.009	**Prob(JB):**	6.50e-09
Kurtosis:	2.755	**Cond. No.**	5.20

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Considerations

- Categorical binary variable as independent.
- Log Transformation of the dependent variable.

Results

H1: Price is strongly impacted by House Renovation.

H0: Price is not strongly impacted by House Renovation.

I'm using a very simple model:

xi = renovation

Yi(price) = B0 + B1xi + e1.

xi = 1 if renovation 0 if no renovation

Looking at the OLS results, not very useful to think of B1 as a slope parameter. Values are 0 and 1, not continous.

The expected price increment when xi = 0, so when no renovation has been taken place.

E(Yi|xi = 0) = B0 + B1(0). = B0

Ln Y^i(price) = 12.9994 %

The expected price increment when Di = 1, so when renovation has been taken place.

E(Yi|xi = 1) = B0 + B1(1).

Ln Y^(price) =12.9994% + 0.2278%

This means that a house that has been renovated command an increase of 0.23% on the price, on average, holding all the other variable costant.

How much of the variation is explained by this results?

R-Squared is 0.009. This means that the 0.9% of the dependent variable Y, in this case price, is explained by the independent variable renovation. Other features should be taken in consideration then.

Is it statistically significant?

Checking the other results:

- p-value is 0.0. It is less than 0.05. This means that I have to reject the H0. I have to reject that renovation feature has no effect on the sale price.
- f-stats is 131.4. so f-stats > 0. It is possible to reject the H0 at 5% level of significance.
- t-value is 11.464 with p-value of O.O which seems to confirm the difference.

Regression Assumptions

Being Boolean, dummy variables are not violating the OLS assumptions. It makes sense check the assumptions when indipendent variables are continous. Another possibilty to perform the analysis would have been to proceed with a logit regression, assuming the price as dummy variable (sold 1, not sold 0) but researchers deem the OLS more reliable. Another choice could have been the one away ANOVA.

Further research

In [37]:
```python
corr_kch = kc_hs.corr().abs()
```

In [38]:
```python
fig, ax=plt.subplots(figsize=(15,10))
fig.suptitle('Houses Features Correlations', fontsize= 20, y=1.0)
heatmap = sns.heatmap(corr_kch, cmap='Blues', annot=True)
```

As it is noticeable the highest scores are related to the size and the condition of the units.

In [39]:
```python
corr_pp = pd.DataFrame(corr_kch.loc[["bedrooms", "bathrooms","sqft_living","sqft_lot",
"condition",
"grade",
"sqft_above",
"sqft_basement",
"zipcode",
"sqft_living15",
"sqft_lot15"],"price"])
corr_pp
```

Out[39]:

	price
bedrooms	0.301992
bathrooms	0.474145
sqft_living	0.648698
sqft_lot	0.077671
floors	0.277432
waterfront	0.084611
condition	0.032624
grade	0.656068
sqft_above	0.548846
sqft_basement	0.256584
zipcode	0.019786
sqft_living15	0.581540
sqft_lot15	0.070292

Correlations Results

Most strongly correlated features that are worthy of further investigations are:

- sqft_living
- sqft_living15
- bathrooms
- grade

Conclusions

For all the stakeholders interested in knowing whether having a house renovated is a lucrative feature, the answer is that the renovation itself, even though it slighly increases the price, is not among the features that considerabily impact the price. Other features are more likely to have a major influence on it such as the living square feet, the evaluation of the condition of the house as per the scale used in Washignton state, the number of bathrooms, and the comparison with other houses in same neighborhood in a 15 mile range. These latter should be deeper investigated.

Further Research Development

The dataset presents two different columns which implicit a scale of evaluation: condition and grade. The difference between seems to be explained by the fact:

- condition - how good the condition is.
- grade - overall grade given to the housing unit, based on King County grading system

In [40]:
```python
kcc_hss5 = kcc_hss[["price","sqft_living", "sqft_living15", "bathrooms", "renovation",
kcc_hss5
```

Out[40]:

	price	sqft_living	sqft_living15	bathrooms	renovation	grade
1	13.195614	2570.0	1690.0	2.25	1	7.0
3	13.311329	1960.0	1360.0	3.00	0	7.0
4	13.142166	1680.0	1800.0	2.00	0	8.0
7	12.583995	1060.0	1650.0	1.50	0	7.0
8	12.343658	1780.0	1780.0	1.00	0	7.0
...	...	...	...	...	...	...
21591	13.071070	1310.0	1330.0	2.50	0	8.0
21592	12.793859	1530.0	1530.0	2.50	0	8.0
21593	12.899220	2310.0	1830.0	2.50	0	8.0
21594	12.904459	1020.0	1020.0	0.75	0	7.0
21596	12.691580	1020.0	1020.0	0.75	0	7.0

15029 rows × 6 columns

In [41]:
```python
kcc_hss5["grade"].unique()
```

Out[41]:
```
array([ 7.,  8.,  9.,  6.,  5., 11., 10.,  4., 12.,  3.])
```

Regression Model

x1 = sqft_living

x2 = sqft_living15

x3 = grade

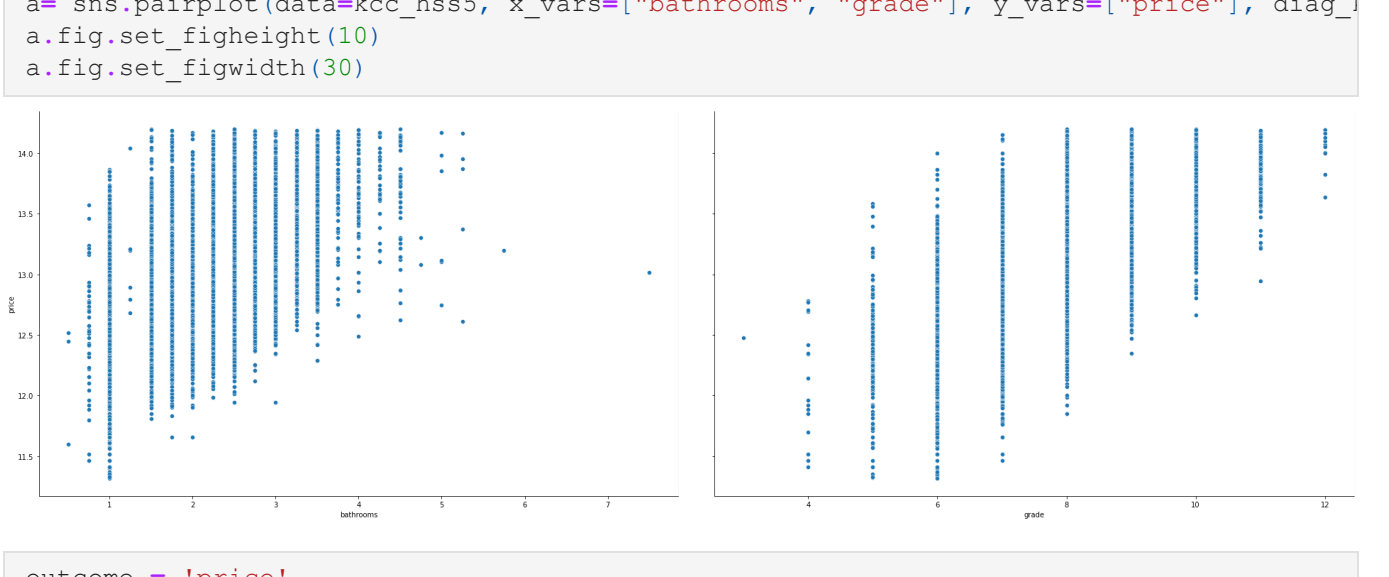
x4 = bathrooms

x5 = renovation

Y(price) = B0 + Bi(x1) + Bi(x2) + Bi(x3) + Bi(x4) + Bi(x5)+ Ei

In [42]:
```python
a= sns.pairplot(data=kcc_hss5, x_vars=['sqft_living','sqft_living15'], y_vars=["price"
a.fig.set_figheight(10)
a.fig.set_figwidth(30)
```

In [43]:
```python
a= sns.pairplot(data=kcc_hss5, x_vars=["bathrooms", "grade"], y_vars=["price"], diag_l
a.fig.set_figheight(10)
a.fig.set_figwidth(30)
```



In [44]:
```python
outcome = 'price'
x_cols = ["sqft_living", "bathrooms", "sqft_living15","grade" ]
predictors = '+'.join(x_cols)
formula = outcome + '~' + predictors
model2 = ols(formula=formula, data=kcc_hss5).fit()
model2.summary()
```

Out[44]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.484
Model:	OLS	**Adj. R-squared:**	0.484
Method:	Least Squares	**F-statistic:**	3521.
Date:	Mon, 20 Jun 2022	**Prob (F-statistic):**	0.00
Time:	22:05:03	**Log-Likelihood:**	-5062.1
No. Observations:	15029	**AIC:**	1.013e+04
Df Residuals:	15024	**BIC:**	1.017e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	11.2960	0.022	506.796	0.000	11.252	11.340
sqft_living	0.0002	6.64e-06	28.288	0.000	0.000	0.000
bathrooms	-0.0146	0.006	-2.564	0.010	-0.026	-0.003
sqft_living15	6.708e-05	6.9e-06	9.725	0.000	5.36e-05	8.06e-05
grade	0.1627	0.004	40.220	0.000	0.155	0.171

Omnibus:	63.444	**Durbin-Watson:**	1.984
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	47.321
Skew:	0.013	**Prob(JB):**	5.30e-11
Kurtosis:	2.726	**Cond. No.**	2.42e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.42e+04. This might indicate that there are strong multicollinearity or other numerical problems.

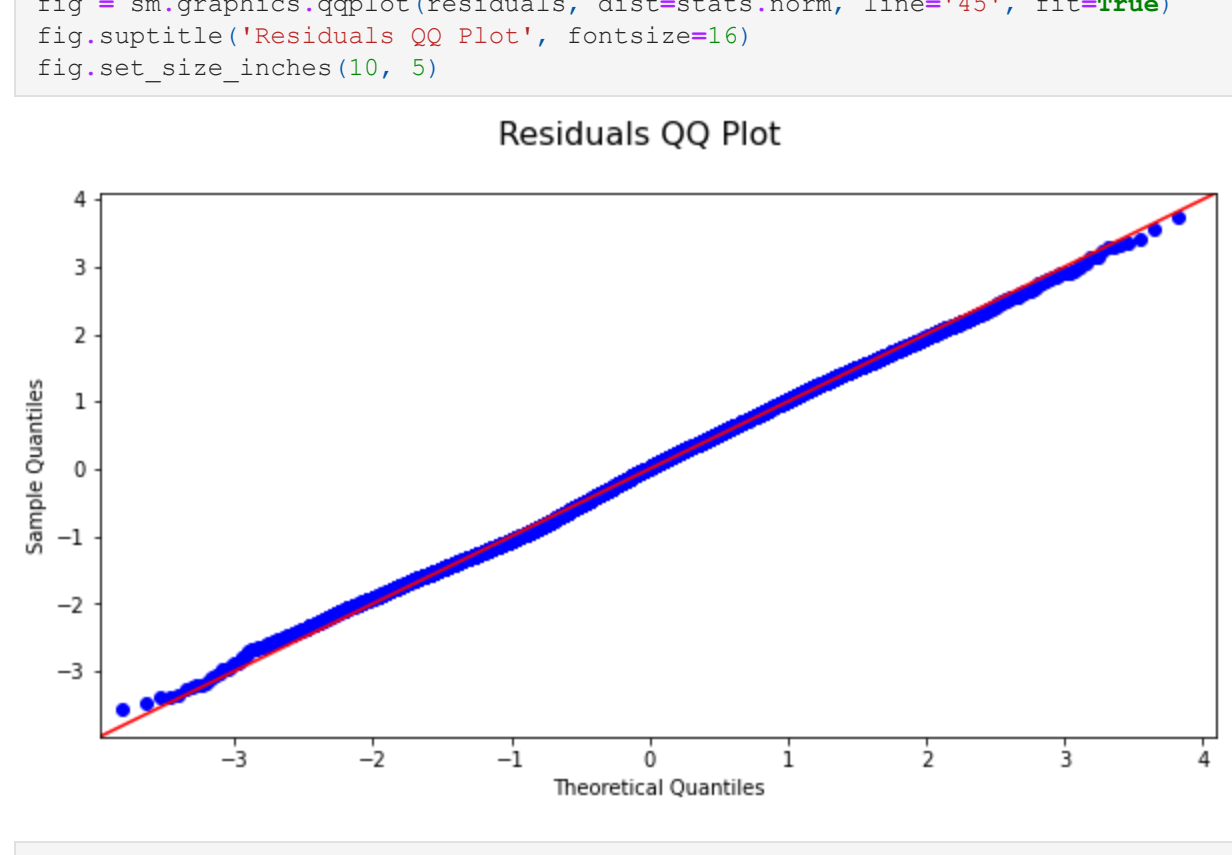
Ln y^(price) = 11.30% +

sqft_living = 0.002

bathrooms = - 0.0146

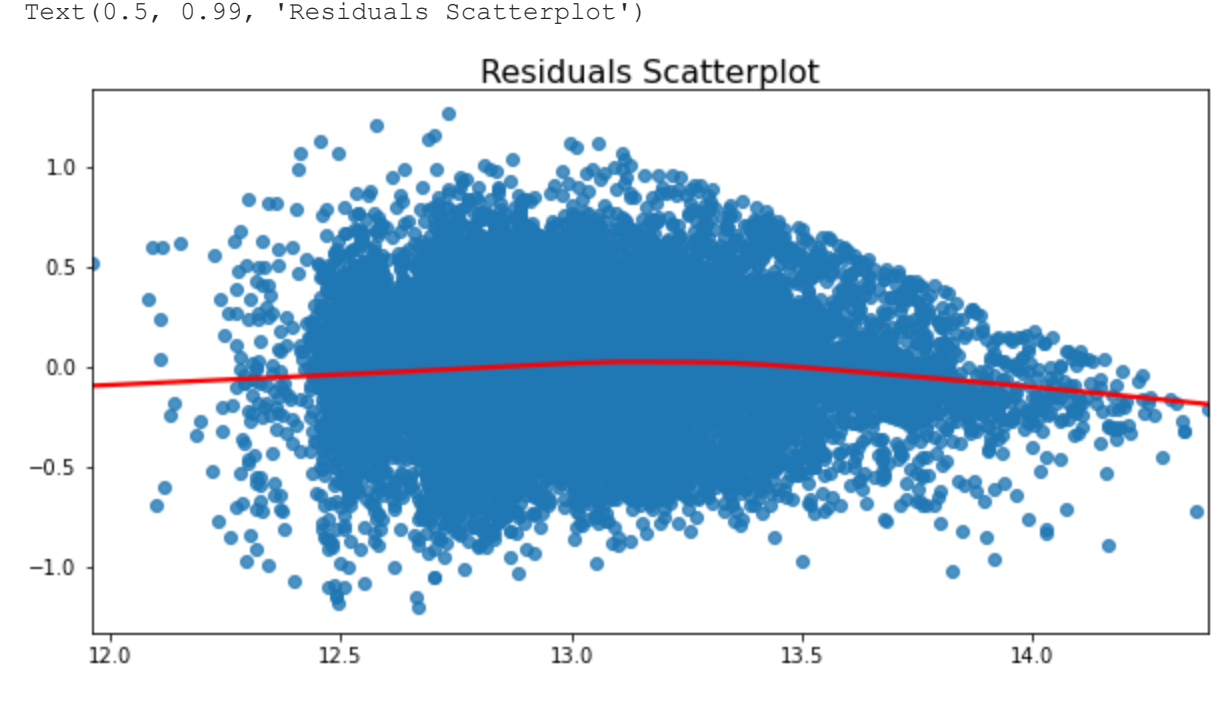
In [45]:
```python
# Residuals plots
```

In [46]:
```python
residuals = model2.resid
fig = sm.graphics.qqplot(residuals, dist=stats.norm, line='45', fit=True)
fig.suptitle('Residuals QQ Plot', fontsize=16)
fig.set_size_inches(10, 5)
```



In [47]:
```python
plt.figure(figsize=(10,5))
sns.regplot(x=model2.predict(), y=model2.resid, lowess=True, line_kws=('color': 'red'
plt.title('Residuals Scatterplot', fontsize=16, y=.99)
```

Out[47]: Text(0.5, 0.99, 'Residuals Scatterplot')



Multilinear regression assumptions

- Linearity. Straight line as model of good fit through the data distribution. The y(price) is maintained in its log transformation. Also, linearity is confirmed by the QQ residuals plot.
- Indipendent Error Terms (Autocorrelation). It occurs with time data. Not applicable in this case.
  - Durbin – Watson (DW) metric. Theoretically, the value must lie between 0 and 4.
    - DW = 2. No autocorrelation.
    - 0 < DW < 2 Positive autocorrelation
    - 2 < DW < 4 indicates negative autocorrelation.
  - Check residual vs time plot for correlated pattern in residual values.
  - Lunjbox Test.
- Normal Error. It is excluded looking at histogram plot and qq plot.
- Multicollinearity. In order to detect it, I examine two metrics:
  - Pearson's correlation.Correlation coefficient between pairs of predictors may lead to identify multicollinearity.
  - Variance Inflation Factor. Interpretation of the metric coefficient:
    - VIF = 1. Absence of multicollinearity.
    - 1< VIF < 2. Absence of a strong multicollnearity.
    - VIF >5, >10. Presence of strong multicollinearity.
- Exogeinity. Giving the R2 squared - highly probable.
- Homoskedasticity (Costant Error Variance). It is confirmed by the residuals scatter plot.

Variance Inflation Factor

In [48]:
```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
```

In [49]:
```python
x_cols = kcc_hss5[["sqft_living", "bathrooms", "sqft_living15","grade" ]]
x_cols
```

Out[49]:

	sqft_living	bathrooms	sqft_living15	grade
1	2570.0	2.25	1690.0	7.0
3	1960.0	3.00	1360.0	7.0
4	1680.0	2.00	1800.0	8.0
7	1060.0	1.50	1650.0	7.0
8	1780.0	1.00	1780.0	7.0
...	...	...	...	...
21591	1310.0	2.50	1330.0	8.0
21592	1530.0	2.50	1530.0	8.0
21593	2310.0	2.50	1830.0	8.0
21594	1020.0	0.75	1020.0	7.0
21596	1020.0	0.75	1020.0	7.0

15029 rows × 4 columns

In [50]:
```python
X = add_constant(x_cols)
pd.Series([variance_inflation_factor(X.values, i)
              for i in range(X.shape[1])],
             index=X.columns)
```

Out[50]:
```
const            64.995638
sqft_living       3.625311
bathrooms         2.205209
sqft_living15     2.527956
grade             2.490079
dtype: float64
```

Results

Considering for each of the variables:

H0: no effect. H1: effect occurs.

Ln Y^(price) =11.27% + 0.0002% - 0.0166% + 7.39% + 0.164% + 0.22%

This means that a house that has been renovated command an increase respectively on the price, on average, holding all the other variable costant. Also, a house that has been renovated command a decrease of 0.0166% respectively on the price, on average, holding all the other variable costant.

How much of the variation is explained by this results?

R-Squared is 0.49. This means that the 49% of the dependent variable Y, in this case price, is explained by these independent variables. Other features should be taken in consideration then.

Is it statistically significant?

Checking the other results:

- p-value is 0.0 and 0.003(bathroms variable). It is less than 0.05. This means that I have to reject the H0.
- f-stats are 2910. so f-stats > 0. It is possible to reject the H0 at 5% level of significance.
- t-values are >0, with p-value of O.O which seems to confirm the difference.

Conclusions

Other considered features were considered more likely to have a major influence on it such as the living square feet, the evaluation of the condition of the house as per the scale used in Washignton state, the number of bathrooms, and the comparison with other houses in same neighborhood in a 15 mile range. Results seem to confirm it but the model seems not be a good model and need various adjustments as only 49% of the IV price seems to be explained by them. The model has to be modified before proceeding to the model training process.

In [ ]: